MobEYE Windows version 1.0 March 2007

Installation and User's Manual

Index:

1. Introduction.....3
2. Windows' specific implementation.....3
3. Requirements.....3
4. Configuration 4
5. Installation.....5
6. How to run MobEYE.....5
7. Setting up a test environment.....5

1. Introduction

MobEYE for windows is a fork from MobEYE V1.2 for Linux, suitable to be run on Windows OS.

It is derived from the MobEYE Linux PDA branch and it works in command line text mode.

It is now possible to build a hybrid MANET, interfacing MobEYE for Windows with MobEYE for Linux (running on laptops and palmtops). Moreover, additional nodes can be included, running only the OLSR routing protocol, under Linux or Windows.

2. Windows' specific implementation

MobEYE for Windows has been kept similar to the original MobEYE for Linux as much as possible. Directory tree and file organization has been maintained, and they remain almost the same.

MobEYE for Windows comes with full source code and Microsoft Visual C dsw project files. Separate executables files will be built for IcpAgent, HttpAgent, CacheManager, InterceptorAgent and the start utility, like for the Linux version.

All components maintain the same interface protocol except for CacheManager: Windows version in fact uses TCP sockets instead of Unix sockets.

Unix configuration file CACHE_SOCKET (string) entry has changed in CACHE_PORT (integer) value in the Windows version configuration file.

MobEYE rely on Posix threads. As Windows does not directly provide support for Posix threads, a third part free library has been used to emulate Posix threads through wrapping proprietary Windows' threads implementation. Posix threads usage has been not changed in MobEYE for Windows' code.

Sockets and file streams handling have been ported to fix issues caused by few differences between the two OSes, while maintaining the same structure and global organization. In particular initialization and error codes check needed changes to port Unix socket code. Care has been paid to avoid problems with different CarrigeReturn coding in payload files.

Directory scan and walking implementation has been changed to maintain the same global semantics while relying on different OS APIs.

Functional structure, functions' names and signatures have been maintained, except for certain configuration file parsing and handling functions, that in Windows version expect an integer for specifying CacheManager TCP port.

Other significant differences include using of Windows' process create _spawnv(...) primitive instead of Posix standard fork() primitive in the start utility, the timeout detection handling mechanism that has been converted to use Windows' timers instead of Posix alarm signals, and InterceptorAgent that does not use anymore sockets, and it is based on WinPcap library.

3. Requirements

To compile MobEYE for Windows from sources you will need Microsoft Visual Studio 6.0, other compilers and development environments might work but they have not been tested. You will also need header and LIB files for WinPcap and Pthreads-w32.

To run MobEYE for Windows you will need:

1. A computer running Microsoft Windows XP.
2. WinPcap library (tested with V 3.1 – see §5).
3. Pthreads-w32 library (tested with V 2.7.0 – see §5).
4. MobEYE for Windows' binaries that comes with this package.
5. A wireless network card supported by Windows OS. To run Interceptor Agent you will need a card for which drivers that allow promiscuous mode exists and are installed.

4. Configuration

Configuration of MobEYE for Windows is mostly the same as for MobEYE for Linux for all Agents except for the InterceptorAgent.

The only difference for other agents is about CACHE_SOCKET option, that in Windows version is changed in CACHE_PORT, expecting an integer number (max 65535) to designate a TCP port number for the cache manager interface.

MobEYE executables will search for a configuration file named "proxy.conf" in their own directory.

The following parameters in the configuration file are supported and required:

- ICP_PORT: This is the UDP port number used for file exchange.
- PROXY_PORT: This is the TCP port number used to achieve communications between browser and proxy (HTTP Agent). This port must be the same in the browser's proxy configuration.
- TIMEOUT: This set the maximum wait time before considering a request failed with no answer.
- CACHE_PORT: This is the TCP port number used to locally communicate with CacheManager.
- INTERFACE_NAME: This is the identifier for the NIC to be used by InterceptorAgent (See below).

This is a sample content for a valid configuration file:

```
ICP_PORT        25150
PROXY_PORT      8080
TIMEOUT         10
CACHE_PORT      15000
CACHE_SIZE      100000
INTERFACE_NAME  0
```

Using of INTERFACE_NAME is essential for successfully run InterceptorAgent: when InterceptorAgent is run, it will scan the configuration file to see if the user has specified a particular NIC from which to drain data packets.

Since users can typically encounter some difficulty to identify NICs on Windows OS, the name "0" could be specified in the configuration file to let Interceptor Agent scan for all the NIC devices available on the system, and let the user choose interactively which one to use.

In this way NICs' names and identificators are printed for each device found.

The user could use this interactive option the first time (s)he runs InterceptorAgent, and then (s)he can copy-and-paste the right NIC identifier in the configuration file for next times (s)he will wish to run InterceptorAgent without having to choose interactively the NIC interface at run time.

This is an example output of InterceptorAgent interactive NIC selection.

```
1. \Device\NPF_GenericDialupAdapter
(Generic dialup adapter)
2. \Device\NPF_{0DD84E0D-26D4-4D10-A196-9D8EEA625EE5}
(Bluetooth PAN Network Adapter NDIS Driver (Microsoft's Packet Scheduler) )
3. \Device\NPF_{7CF529A3-CD0B-473E-AE5F-4B47E464C87E}
(Broadcom 440x 10/100 Integrated Controller)
4. \Device\NPF_{83EDB289-9C82-4055-9B19-4BFD8C995035}
(Intel(R) PRO/Wireless 2200BG Network Connection)
Selezionare interfaccia di rete(1-4):
```

In this example selecting interface “4” will cause InterceptorAgent to use the Intel mini-pci card present in the system to listen to packets until next program run.

To make InterceptorAgent using this card by default without boring with the interactive selection the configuration file can be adjusted setting the INTERFACE_NAME parameter as follows

```
INTERFACE_NAME = \Device\NPF_{83EDB289-9C82-4055-9B19-4BFD8C995035}
```

Please note that some Windows’ WiFi network card drivers will refuse to pass any packet to higher layer when set in promiscuous mode. This is effectively a limitation of those drivers (and sometimes NIC firmware) and it will result in InterceptorAgent inability to receive any packet, without giving any specific error warnings.

5. Installation

To install MobEYE it is enought to copy the whole MobEYE binary directory (including subdirectories) on the target directory you wish to install MobEYE to, and to make sure libraries' DLL files for pthread-w32 and WinPcap are in the right place (usually “%SystemRoot%\system32” that means in most cases “c:\windows\system32”, or the MobEYE executables directory).

The WinPcap’s installer will do this by itself, while for user’s convenience binary pthread-w32 DLL library has been already included in the MobEYE binary directory.

So one will only need to install WinPcap before running MobEYE for Windows.

WinPcap can be downloaded from http://www.winpcap.org while pthread-w32 can be found at http://sourceware.org/pthreads-win32/

6. How to run MobEYE

To run MobEYE, after you installed and configured it, just run the program “start” that is in the binary directory.

Please pay attention that the name “start” is also used as a Windows command, so you can run it from the GUI or, if you wish to run from the command prompt, rename it, or make sure Windows known which file you want to run by specifying full PATH.

7. Setting up a test environment

A demo environment could be made by few hosts, each one running MobEYE and Olsr.

The first thing to do is to configure the OS to participate in the MANET network (in other words putting the NIC in IBSS mode and correctly configuring network and IP parameters).

In this example we will use “MobTest” as network name, and the 192.168.0.0/24 private subnet IP addresses. We will also assume the NIC on Linux OS is named “wlan0”; this is really dependant by your distribution and the wireless network driver in use.

For Linux OS, as root, run the following commands:

```
iwconfig wlan0 mode ad-hoc
iwconfig wlan0 essid ModTest
```

Certain drivers might also require the channel to be set. It can be done with the following command:

```
iwconfig wlan0 channel X
```

The “X” can be any channel allowed in your country. Typically the default used channel for this kind of networks is channel number 11.

At this point, to start the network, just assign an IP address to the NIC. The following command will assign the address 192.168.0.1 (that is the first valid address in the subnet we are considering in this example) to the NIC:

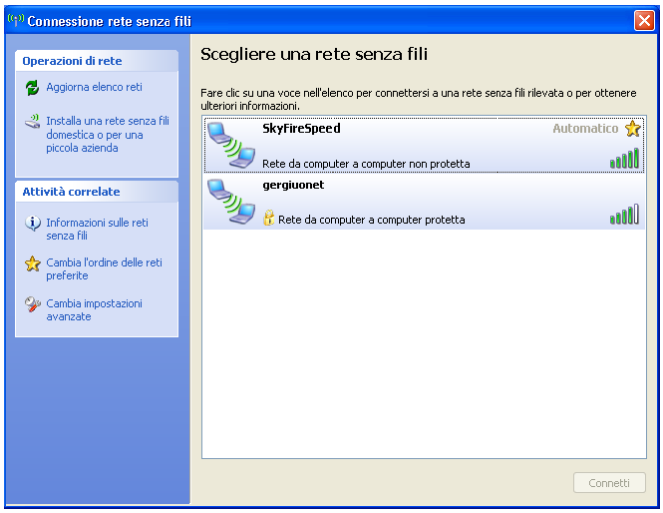
```
ifconfig wlan0 192.168.0.1
```

Now the host will create an IBSS cell with a randomly generated BSSID and “ModTest” ESSID, or it will join an existent cell with “ModTest” ESSID.

Currently all Linux wireless drivers (with maybe the exception of certain rare proprietary ones) use Wireless Extensions, that means all of them can be configured using the above commands; but since device drivers are developed by several people around the world, the semantics of those command might be not always perfectly consistent. For this reason reading the documentation of the specific driver in use is suggested (especially if the above standard procedure fails).

The new Linux mac80211 layer (contained in the wireless-dev git tree) and the new configuration API should hopefully solve this problem definitely, but at the moment it is still in a quite early stage, so it is not possible to do serious tests with them and provide tested instruction about how to configure them (but they should be still compatible with Wireless Extension).

On Windows OS (assuming SP2 is installed) creating or joining an IBBS (ad-hoc) network is rather simple: left click on the wireless NIC icon ( ) on the systray bar; the network configuration window should appear.

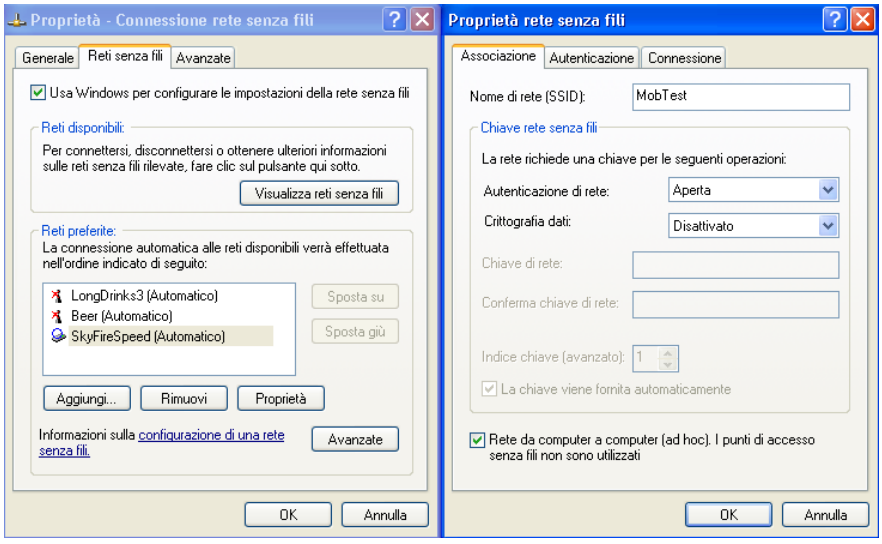

Connessione rete senza fili
Operazioni di rete
Aggiorna elenco reti
Installa una rete senza fili domestica o per una piccola azienda
Attività correlate
Informazioni sulle reti senza fili
Cambia l'ordine delle reti preferite
Cambia impostazioni avanzate
Scegliere una rete senza fili
Fare clic su una voce nell'elenco per connettersi a una rete senza fili rilevata o per ottenere ulteriori informazioni.
SkyFireSpeed
Automatico
Rete da computer a computer non protetta
gergiuonet
Rete da computer a computer protetta
Connetti

By left clicking the wireless network card the NIC configuration dialog will appear

Here double-click an existent IBSS network to join it, or click the “Change advanced settings” link on the left bar to enter in the NIC properties dialog.

Here left click the “wireless networks” tab and click the “add” button.

In the dialog box that should pop-up enter “MobTest” in the SSID text box and thick the “peer to peer network (ad-hoc)” radio button, then confirm with okays.

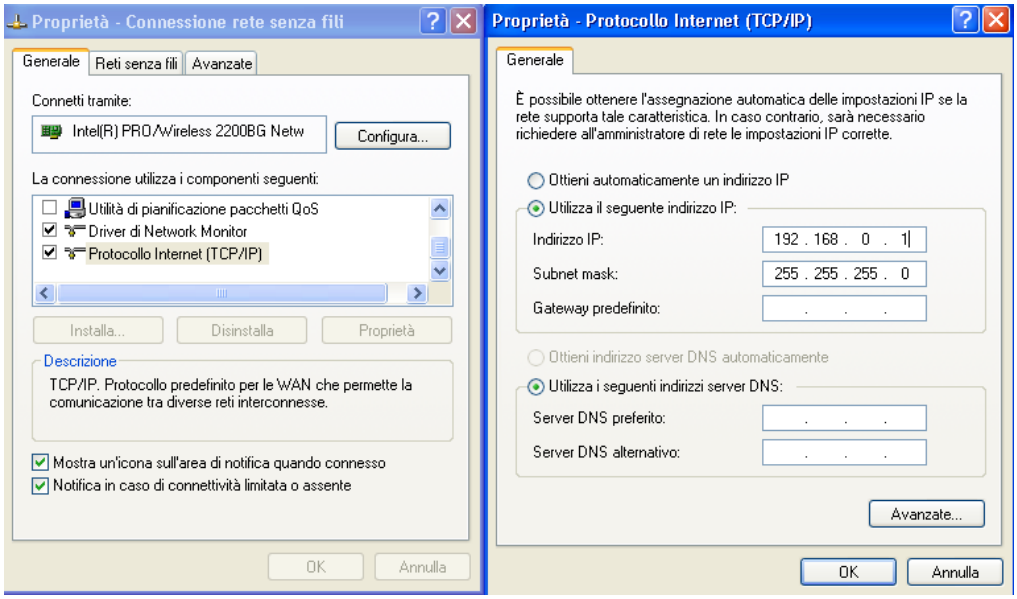


Clicking “Change advanced settings” and the “Add” button will result in this two dialog boxes

At this stage (once a network has been joined or created) an IP address has still to be assigned to the NIC. To do this open again the NIC proprieties dialog as explained above, select “Internet Protocol (TCP/IP)” in the list box, and click “propriety”.

Select “use fixed IP address” then fill the first two field (“Ip address” and “subnet mask”) and confirm with okays.

The subnet mask in this example must be 255.255.255.0, while for the IP address any valid and not-used IP address in the subnet is acceptable (for example 192.168.0.2).


Proprietà - Connessione rete senza fili
Generale
Reti senza fili
Avanzate
Connetti tramite:
Intel(R) PRO/Wireless 2200BG Netw
Configura...
La connessione utilizza i componenti seguenti:
Utilità di pianificazione pacchetti QoS
Driver di Network Monitor
Protocollo Internet (TCP/IP)
Installa...
Disinstalla
Proprietà
Descrizione
TCP/IP. Protocollo predefinito per le WAN che permette la comunicazione tra diverse reti interconnesse.
Mostra un'icona sull'area di notifica quando connesso
Notifica in caso di connettività limitata o assente
OK
Annulla
Proprietà - Protocollo Internet (TCP/IP)
Generale
È possibile ottenere l'assegnazione automatica delle impostazioni IP se la rete supporta tale caratteristica. In caso contrario, sarà necessario richiedere all'amministratore di rete le impostazioni IP corrette.
Ottieni automaticamente un indirizzo IP
Utilizza il seguente indirizzo IP:
Indirizzo IP:
192 . 168 . 0 . 1
Subnet mask:
255 . 255 . 255 . 0
Gateway predefinito:
Ottieni indirizzo server DNS automaticamente
Utilizza i seguenti indirizzi server DNS:
Server DNS preferito:
Server DNS alternativo:
Avanzate...
OK
Annulla

Those two dialog boxes will allow the IP configuration

On each host also Olsr routing protocol is needed.

Installation is quite simple: For Linux OS each distribution offers different packages and different tools to get and install packages, however if compilers and standard development tools are installed on the system, compiling Olsr from source should be always possible.

To extract the source tarball the following command can be used (the tarball and directory name might differ because of the version number)

```
tar xjvf olsrs-0.4.10.tar.bz2
```

Compilation can be run using the standard make command as follows

```
cd olsrd-0.4.10
make
```

Then, as root, to complete the installation run

```
make install
```

For Windows OS an installation program will automatically guide the user through all the step required to complete the installation, just run it.

Attention should be paid to correctly configure Olsr consistently on all hosts.

To better understand Olsr's configuration options and details reading the Olsr user manual is recommended.